Package: tidynorm (via r-universe)

October 26, 2026

Type Package

Title Tools for Tidy Vowel Normalization

Version 0.5.0

Description An implementation of tidy speaker vowel normalization. This includes generic functions for defining new normalization methods for points, formant tracks, and Discrete Cosine Transform coefficients, as well as convenience functions implementing established normalization methods. References for the implemented methods are: Johnson, Keith (2020) <doi:10.5334/labphon.196> Lobanov, Boris (1971) <doi:10.1121/1.1912396> Nearey, Terrance M. (1978) <https://sites.ualberta.ca/~tnearey/Nearey1978_compressed.pdf> Syrdal, Ann K., and Gopal, H. S. (1986) <doi:10.1121/1.393381> Watt, Dominic, and Fabricius, Anne (2002) <https://www.latl.leeds.ac.uk/article/evaluation-of-a-technique-for-improving-the-mapping-of-multiple-speakers-vowel-spaces-in-the-f1-

License GPL (>= 3)

Encoding UTF-8

LazyData true

Imports checkmate, cli, dplyr, glue, options, purrr, Rcpp, rlang, stringr, tidyr, tidyselect

URL https://jofrhwld.github.io/tidynorm/, https://github.com/JoFrhwld/tidynorm

Depends R (>= 4.1)

Suggests ggdensity, ggplot2, knitr, magick, quarto, reticulate, rmarkdown, testthat (>= 3.0.0), tibble

Config/testthat/edition 3

Config/testthat/parallel true

VignetteBuilder quarto

LinkingTo Rcpp, RcppArmadillo

BugReports https://github.com/JoFrhwld/tidynorm/issues

Config/roxygen2/version 8.1.1

NeedsCompilation yes

Author Josef Fruehwald [cre, aut, cph]

Maintainer Josef Fruehwald <JoFrhwld@gmail.com>

Repository https://cran.r-universe.dev

Date/Publication 2026-10-26 17:50:02 UTC

RemoteUrl https://github.com/cran/tidynorm

RemoteRef HEAD

RemoteSha 2bd8e519b1cdf70c6748e9fc06cccb5ef0e84ace

Contents

bark_to_hz . . . 3
check_norm . . . 4
dct_basis . . . 4
hz_to_bark . . . 5
hz_to_mel . . . 6
idct_accel . . . 7
idct_rate . . . 8
mel_to_hz . . . 9
norm_barkz . . . 10
norm_dct_barkz . . . 12
norm_dct_deltaF . . . 14
norm_dct_generic . . . 17
norm_dct_lobanov . . . 20
norm_dct_nearey . . . 23
norm_dct_wattfab . . . 26
norm_deltaF . . . 28
norm_generic . . . 31
norm_lobanov . . . 33
norm_nearey . . . 35
norm_track_barkz . . . 38
norm_track_deltaF . . . 41
norm_track_generic . . . 44
norm_track_lobanov . . . 49
norm_track_nearey . . . 52
norm_track_wattfab . . . 55
norm_wattfab . . . 58
options . . . 61
reframe_with_dct . . . 62
reframe_with_dct_smooth . . . 63
reframe_with_idct . . . 65
speaker_data . . . 68
speaker_tracks . . . 68
tidynorm_options . . . 69

Index **71**

`bark_to_hz` *Bark to Hz*

Description

Converts bark to Hz

Usage

```
bark_to_hz(bark)
```

Arguments

`bark`	Frequency in Bark

Details

$$\hat{b} = \begin{cases} \frac{b-0.3}{0.85} & \text{if } b < 2 \\ \frac{b+4.422}{1.22} & \text{if } b > 20.1 \\ b & \text{otherwise} \end{cases}$$

$$hz = 1960 \frac{\hat{b} + 0.53}{26.28 - \hat{b}}$$

Value

A vector of Hz scaled values

References

Traunmüller, H. (1990). Analytical expressions for the tonotopic sensory scale. The Journal of the Acoustical Society of America, 88(1), 97–100. doi:10.1121/1.399849

Examples

```
bark <- seq(1.5, 13, length = 100)
hz <- bark_to_hz(bark)
plot(bark, hz)
```

check_norm *Check Normalization Procedures*

Description

check_norm() will generate a message with information about which normalization procedures have been applied to the data.

Usage

```
check_norm(.data)
```

Arguments

.data	A data frame produced by a tidynorm function.

Value

This only prints an info message.

Examples

```
speaker_norm <- speaker_data |>
  norm_nearey(
    F1:F3,
    .by = speaker,
    .silent = TRUE
  )

check_norm(speaker_norm)
```

dct_basis *DCT Basis*

Description

The Discrete Cosine Transform basis functions

Usage

```
dct_basis(n, k)
```

Arguments

n	The length of the basis.
k	The number of basis functions.

Details

This function will generate the DCT basis functions.

Value

A $n \times k$ matrix

Examples

```
basis <- dct_basis(100, 5)
matplot(basis, type = "l", lty = 1)
```

`hz_to_bark` *Hz to Bark*

Description

Converts Hz to Bark

Usage

```
hz_to_bark(hz)
```

Arguments

`hz`	Frequency in Hz

Details

$$\hat{b} = \frac{26.81hz}{1960 + hz} - 0.53$$

$$b = \begin{cases} \hat{b} + 0.15(2 - \hat{b}) & \text{if } \hat{b} < 2 \\ \hat{b} + 0.22(\hat{b} - 20.1) & \text{if } \hat{b} > 20.1 \\ \hat{b} & \text{otherwise} \end{cases}$$

Value

A vector of bark scaled values

References

Traunmüller, H. (1990). Analytical expressions for the tonotopic sensory scale. The Journal of the Acoustical Society of America, 88(1), 97–100. doi:10.1121/1.399849

Examples

```
hz <- seq(150, 2000, length = 100)
bark <- hz_to_bark(hz)
plot(hz, bark)
```

`hz_to_mel` *Hz to Mel*

Description

Convert Hz to Mel

Usage

```
hz_to_mel(hz, htk = FALSE)
```

Arguments

`hz`	Numeric values in Hz
`htk`	Whether or not to use the HTK formula

Details

This is a direct re-implementation of the `hz_to_mel` function from the librosa library.

The default method is to use the method due to Slaney (1998), which is linear below 1000Hz, and logarithmic above.

If `htk=TRUE`, the method from HTK, due to O'Shaughnessy (1987) is used.

Value

A numeric vector of Mel values

References

McFee, B., C. Raffel, D. Liang, D. PW Ellis, M. McVicar, E. Battenberg, and O. Nieto. librosa: Audio and music signal analysis in python. In Proceedings of the 14th python in science conference, pp. 18-25.

O'Shaughnessy, D (1987). Speech communication: human and machine. Addison-Wesley. p. 150. ISBN 978-0-201-16520-3.

Slaney, M. (1998) Auditory Toolbox: A MATLAB Toolbox for Auditory Modeling Work. Technical Report, version 2, Interval Research Corporation.

Examples

```
hz_to_mel(c(500, 1000, 2000, 3000))
```

idct_accel *Inverse Discrete Cosine Transform Acceleration*

Description

The second derivative of the Inverse DCT

Usage

```
idct_accel(y, n = length(y))
```

Arguments

y	DCT coefficients
n	The desired length of the idct

Details

Returns the second derivative (acceleration) of the Inverse DCT (see dct for more details).

$$\frac{\delta^2 x_j}{\delta j^2} = -2\left(\frac{\pi k}{J}\right)^2 \sum_{k=1}^{N-1} y_k \cos\left(\frac{\pi k(2j+1)}{2J}\right)$$

Value

A vector with the second derivative of the inverse DCT

Examples

```
x <- seq(0, 1, length = 10)
y <- 5 + x + (2 * (x^2)) + (-2 * (x^4))

dct_coefs <- dct(y)
y_accel <- idct_accel(dct_coefs)

plot(y)
plot(y_accel)
```

idct_rate *Inverse Discrete Cosine Transform Rate*

Description

The first derivative of the Inverse DCT

Usage

```
idct_rate(y, n = length(y))
```

Arguments

y	DCT coefficients
n	The desired length of the idct

Details

Returns the first derivative (rate of change) of the Inverse DCT (see dct for more details).

$$\frac{\delta x_j}{\delta j} = -2\frac{\pi k}{J}\sum_{k=1}^{N-1} y_k \sin\left(\frac{\pi k(2j+1)}{2J}\right)$$

Value

A vector with the first derivative of the inverse DCT

Examples

```
x <- seq(0, 1, length = 10)
y <- 5 + x + (2 * (x^2)) + (-2 * (x^4))

dct_coefs <- dct(y)
y_rate <- idct_rate(dct_coefs)

plot(y)
plot(y_rate)
```

`mel_to_hz` *Mel to Hz*

Description

Convert Mel to Hz

Usage

```
mel_to_hz(mel, htk = FALSE)
```

Arguments

`mel`	Numeric values in Hz
`htk`	Whether or not to use the HTK formula

Details

This is a direct re-implementation of the `hz_to_mel` function from the librosa library.

The default method is to use the method due to Slaney (1998), which is linear below 1000Hz, and logarithmic above.

If `htk=TRUE`, the method from HTK, due to O'Shaughnessy (1987) is used.

Value

A numeric vector of Hz values

References

McFee, B., C. Raffel, D. Liang, D. PW Ellis, M. McVicar, E. Battenberg, and O. Nieto. librosa: Audio and music signal analysis in python. In Proceedings of the 14th python in science conference, pp. 18-25.

O'Shaughnessy, D (1987). Speech communication: human and machine. Addison-Wesley. p. 150. ISBN 978-0-201-16520-3.

Slaney, M. (1998) Auditory Toolbox: A MATLAB Toolbox for Auditory Modeling Work. Technical Report, version 2, Interval Research Corporation.

Examples

```
mel_to_hz(c(7.5, 15, 25, 31))
```

norm_barkz *Bark Difference Normalize*

Description

Bark Difference Normalize

Usage

```
norm_barkz(
  .data,
  ...,
  .by = NULL,
  .drop_orig = FALSE,
  .keep_params = FALSE,
  .names = "{.formant}_bz",
  .silent = opt("tidynorm.silent")
)
```

Arguments

.data	A data frame containing vowel formant data
...	<tidy-select> One or more unquoted expressions separated by commas. These should target the vowel formant data columns.
.by	<tidy-select> A selection of columns to group by. Typically a column of speaker IDs.
.drop_orig	Whether or not to drop the original formant data columns.
.keep_params	Whether or not to keep the Location (*_.L) and Scale (*_.S) normalization parameters
.names	A glue::glue() expression for naming the normalized data columns. The "{.formant}" portion corresponds to the name of the original formant columns.
.silent	Suppress normalization information messages when running a norm_*() function. (Defaults to FALSE, overwritable using option 'tidynorm.silent' or environment variable 'R_TIDYNORM_TIDYNORM_SILENT')

Details

This is a within-token normalization technique. First all formants are converted to Bark (see hz_to_bark), then, within each token, F3 is subtracted from F1 and F2.

$$\hat{F}_{ij} = F_{ij} - L_j$$

$$L_j = F_{3j}$$

Pre-aggregation:

As a vowel-intrinsic normalization technique, the Bark differenc methid is incompatible with pre-aggregation.

Value

A data frame of Bark Difference normalized formant values

References

Syrdal, A. K., & Gopal, H. S. (1986). A perceptual model of vowel recognition based on the auditory representation of American English vowels. The Journal of the Acoustical Society of America, 79(4), 1086–1100. doi:10.1121/1.393381

Examples

```
library(tidynorm)
ggplot2_inst <- require(ggplot2)

speaker_data_barkz <- speaker_data |>
  norm_barkz(
    F1:F3,
    .by = speaker,
    .names = "{.formant}_bz"
  )

if (ggplot2_inst) {
  ggplot(
    speaker_data_barkz,
    aes(
      F2_bz,
      F1_bz,
      color = speaker
    )
  ) +
    stat_density_2d(
      bins = 4
    ) +
    scale_color_brewer(
      palette = "Dark2"
    ) +
    scale_x_reverse() +
    scale_y_reverse() +
    coord_fixed()
}
```

`norm_dct_barkz` *Bark Difference DCT Normalization*

Description

Bark Difference DCT Normalization

Usage

```
norm_dct_barkz(
  .data,
  ...,
  .token_id_col,
  .by = NULL,
  .param_col = NULL,
  .drop_orig = FALSE,
  .names = "{.formant}_bz",
  .silent = opt("tidynorm.silent")
)
```

Arguments

`.data`	A data frame containing vowel formant data
`...`	<tidy-select> One or more unquoted expressions separated by commas. These should target the vowel formant data columns.
`.token_id_col`	<data-masking> A column that identifies token ids.
`.by`	<tidy-select> A selection of columns to group by. Typically a column of speaker IDs.
`.param_col`	A column identifying the DCT parameter number.
`.drop_orig`	Should the originally targeted columns be dropped.
`.names`	A `glue::glue()` expression for naming the normalized data columns. The `"{.formant}"` portion corresponds to the name of the original formant columns.
`.silent`	Suppress normalization information messages when running a `norm_*()` function. (Defaults to `FALSE`, overwritable using option 'tidynorm.silent' or environment variable 'R_TIDYNORM_TIDYNORM_SILENT')

Details

Important: This function assumes that the DCT coefficients were estimated over bark-transformed formant values.

This is a within-token normalization technique. First all formants are converted to Bark (see hz_to_bark), then, within each token, F3 is subtracted from F1 and F2.

$$\hat{F}_{ij} = F_{ij} - L_j$$

$$L_j = F_{3j}$$

Value

A data frame of normalized DCT parameters.

A data frame of Back Difference normalized dct coefficients.

References

Syrdal, A. K., & Gopal, H. S. (1986). A perceptual model of vowel recognition based on the auditory representation of American English vowels. The Journal of the Acoustical Society of America, 79(4), 1086–1100. doi:10.1121/1.393381

Examples

```
library(tidynorm)
library(dplyr)
ggplot2_inst <- require(ggplot2)

speaker_dct <- speaker_tracks |>
  mutate(
    across(F1:F3, hz_to_bark)
  ) |>
  reframe_with_dct(
    F1:F3,
    .by = speaker,
    .token_id_col = id,
    .time_col = t
  )

# Normalize DCT coefficients
speaker_dct_norm <- speaker_dct |>
  norm_dct_barkz(
    F1:F3,
    .by = speaker,
    .token_id_col = id,
    .param_col = .param
  )

# Apply average and apply inverse dct
# to plot tracks
track_norm_means <- speaker_dct_norm |>
  summarise(
    .by = c(speaker, vowel, .param),
    across(
      ends_with("_bz"),
      mean
    )
  ) |>
  reframe_with_idct(
    ends_with("_bz"),
    .by = speaker,
    .token_id_col = vowel,
    .param_col = .param
```

```
  )


if (ggplot2_inst) {
  track_norm_means |>
    ggplot(
      aes(F2_bz, F1_bz, color = speaker)
    ) +
    geom_path(
      aes(
        group = interaction(speaker, vowel)
      )
    ) +
    scale_x_reverse() +
    scale_y_reverse() +
    scale_color_brewer(palette = "Dark2") +
    coord_fixed()
}
```

`norm_dct_deltaF` *Delta F DCT Normalization*

Description

Delta F DCT Normalization

Usage

```
norm_dct_deltaF(
  .data,
  ...,
  .token_id_col,
  .by = NULL,
  .agg_by = NULL,
  .agg_fun = function(x) mean(x, na.rm = T),
  .param_col = NULL,
  .drop_orig = FALSE,
  .names = "{.formant}_df",
  .silent = opt("tidynorm.silent")
)
```

Arguments

`.data`	A data frame containing vowel formant data
`...`	<tidy-select> One or more unquoted expressions separated by commas. These should target the vowel formant data columns.
`.token_id_col`	<data-masking> A column that identifies token ids.

.by	<tidy-select> A selection of columns to group by. Typically a column of speaker IDs.
.agg_by	<tidy-select> A selection of columns to to pre-aggregate by. Typically a column of vowel category labels.
.agg_fun	A function to use to pre-aggregate values.
.param_col	A column identifying the DCT parameter number.
.drop_orig	Should the originally targeted columns be dropped.
.names	A glue::glue() expression for naming the normalized data columns. The "{.formant}" portion corresponds to the name of the original formant columns.
.silent	Suppress normalization information messages when running a norm_*() function. (Defaults to FALSE, overwritable using option 'tidynorm.silent' or environment variable 'R_TIDYNORM_TIDYNORM_SILENT')

Details

$$\hat{F}_{ij} = \frac{F_{ij}}{S}$$

$$S = \frac{1}{MN} \sum_{i=1}^{M} \sum_{j=1}^{N} \frac{F_{ij}}{i - 0.5}$$

Where

- $\hat{F}$ is the normalized formant
- i is the formant number
- j is the token number

Pre-aggregation:

To address imbalanced data across vowel classes, you can pre-aggregate the zeroth DCT coefficients by passing vowel class column(s) to .agg_by, and and an aggregating function to .agg_fun. The aggregation will be done after the DCT transform is applied, and will be done either formant intrinsically or formant extrinsically based on the setting passed to .by_formant.

Value

A data frame of Delta F normalized DCT coefficients.

References

Johnson, K. (2020). The ΔF method of vocal tract length normalization for vowels. Laboratory Phonology: Journal of the Association for Laboratory Phonology, 11(1), Article 1. doi:10.5334/labphon.196

Examples

```
library(tidynorm)
library(dplyr)
ggplot2_inst <- require(ggplot2)

speaker_dct <- speaker_tracks |>
  reframe_with_dct(
    F1:F3,
    .by = speaker,
    .token_id_col = id,
    .time_col = t
  )

# Normalize DCT coefficients
speaker_dct_norm <- speaker_dct |>
  norm_dct_deltaF(
    F1:F3,
    .by = speaker,
    .token_id_col = id,
    .param_col = .param
  )

# Apply average and apply inverse dct
# to plot tracks
track_norm_means <- speaker_dct_norm |>
  summarise(
    .by = c(speaker, vowel, .param),
    across(
      ends_with("_df"),
      mean
    )
  ) |>
  reframe_with_idct(
    ends_with("_df"),
    .by = speaker,
    .token_id_col = vowel,
    .param_col = .param
  )


if (ggplot2_inst) {
  track_norm_means |>
    ggplot(
      aes(F2_df, F1_df, color = speaker)
    ) +
    geom_path(
      aes(
        group = interaction(speaker, vowel)
      )
    ) +
    scale_x_reverse() +
    scale_y_reverse() +
```

```
    scale_color_brewer(palette = "Dark2") +
    coord_fixed()
}
```

`norm_dct_generic` *Generic Formant DCT Normalization Procedure*

Description

Generic Formant DCT Normalization Procedure

Usage

```
norm_dct_generic(
  .data,
  ...,
  .token_id_col,
  .by = NULL,
  .agg_by = NULL,
  .agg_fun = function(x) mean(x, na.rm = T),
  .param_col = NULL,
  .L = 0,
  .S = 1/sqrt(2),
  .by_formant = FALSE,
  .by_token = FALSE,
  .names = "{.formant}_n",
  .silent = opt("tidynorm.silent"),
  .drop_orig = FALSE,
  .call = caller_env()
)
```

Arguments

`.data`	A data frame of formant DCT coefficients
`...`	<tidy-select> One or more unquoted expressions separated by commas. These should target the vowel formant data columns.
`.token_id_col`	<data-masking> A column that identifies token ids.
`.by`	<tidy-select> A selection of columns to group by. Typically a column of speaker IDs.
`.agg_by`	<tidy-select> A selection of columns to to pre-aggregate by. Typically a column of vowel category labels.
`.agg_fun`	A function to use to pre-aggregate values.
`.param_col`	A column identifying the DCT parameter number.
`.L`	An expression defining the location parameter. See Details for more information.

`.S`	An expression defining the scale parameter. See Details for more information.
`.by_formant`	Whether or not the normalization method is formant intrinsic.
`.by_token`	Whether or not the normalization method is token intrinsic
`.names`	A `glue::glue()` expression for naming the normalized data columns. The `"{.formant}"` portion corresponds to the name of the original formant columns.
`.silent`	Suppress normalization information messages when running a `norm_*()` function. (Defaults to `FALSE`, overwritable using option 'tidynorm.silent' or environment variable 'R_TIDYNORM_TIDYNORM_SILENT')
`.drop_orig`	Should the originally targeted columns be dropped.
`.call`	Used for internal purposes.

Details

The following `norm_dct_*` procedures were built on top of `norm_dct_generic()`.

- norm_dct_lobanov
- norm_dct_nearey
- norm_dct_deltaF
- norm_dct_wattfab
- norm_dct_barkz

Normalizing DCT Coefficients:

This will normalize vowel formant data that has already had the Discrete Cosine Transform applied (see dct) with the following procedure:

1. Location `.L` and Scale `.S` expressions will be used to summarize the zeroth DCT coefficients.
2. These location and scale will be used to normalize the DCT coefficients.

Location and Scale expressions:
norm_dct_generic normalizes DCT coefficients directly. If F_k is the kth DCT coefficient the normalization procedure is

$$\hat{F}_k = \frac{F_k - L'}{\sqrt{2}S}$$

$$L' = \begin{cases} L & \text{for } k = 0 \\ 0 & \text{for } k > 0 \end{cases}$$

Rather than requiring users to remember to multiply expressions for S by $\sqrt{2}$, this is done by norm_dct_generic itself, to allow greater parallelism with how norm_generic works.
Note: If you want to scale values by a constant in the normalization, you'll need to divide the constant by `sqrt(2)`.
The expressions for calculating L and S can be passed to `.L` and `.S`, respectively. Available values for these expressions are

`.formant` The original formant value

`.formant_num` The number of the formant. (e.g. 1 for F1, 2 for F2 etc)

Along with any data columns from your original data.

Identifying tokens:
DCT normalization requires identifying individual tokens, so there must be a column that uniquely identifies (or, in combination with a `.by` grouping, uniquely identifies) each individual token. This column should be passed to `.token_id_col`.

Pre-aggregation:
To address imbalanced data across vowel classes, you can pre-aggregate the zeroth DCT coefficients by passing vowel class column(s) to `.agg_by`, and and an aggregating function to `.agg_fun`. The aggregation will be done after the DCT transform is applied, and will be done either formant intrinsically or formant extrinsically based on the setting passed to `.by_formant`.

Value

A data frame of normalized DCT coefficients.

Examples

```
library(tidynorm)
library(dplyr)
ggplot2_inst <- require(ggplot2)

track_subset <- speaker_tracks |>
  filter(
    .by = c(speaker, id),
    if_all(
      F1:F3,
      .fns = \(x) mean(is.finite(x)) > 0.9
    ),
    row_number() %% 2 == 1
  )

track_dcts <- track_subset |>
  reframe_with_dct(
    F1:F3,
    .by = speaker,
    .token_id_col = id,
    .time_col = t,
    .order = 3
  )

track_norm <- track_dcts |>
  norm_dct_generic(
    F1:F3,
    .token_id_col = id,
    .by = speaker,
    .by_formant = TRUE,
    .L = median(.formant, na.rm = TRUE),
    .S = mad(.formant, na.rm = TRUE),
    .param_col = .param,
    .drop_orig = TRUE,
    .names = "{.formant}_mad"
  )
```

```
head(track_norm)

full_tracks <- track_norm |>
  summarise(
    .by = c(speaker, vowel, .param),
    across(
      F1_mad:F3_mad,
      mean
    )
  ) |>
  reframe_with_idct(
    F1_mad:F3_mad,
    .by = c(speaker, vowel),
    .param_col = .param
  )

head(full_tracks)

if (ggplot2_inst) {
  ggplot(
    full_tracks,
    aes(F2_mad, F1_mad, color = speaker)
  ) +
    geom_path(
      aes(group = interaction(speaker, vowel))
    ) +
    scale_y_reverse() +
    scale_x_reverse() +
    scale_color_brewer(palette = "Dark2") +
    coord_fixed()
}
```

`norm_dct_lobanov` *Lobanov DCT Normalization*

Description

Lobanov DCT Normalization

Usage

```
norm_dct_lobanov(
  .data,
  ...,
  .token_id_col,
  .by = NULL,
  .agg_by = NULL,
  .agg_fun = function(x) mean(x, na.rm = T),
  .param_col = NULL,
  .names = "{.formant}_z",
```

```
  .silent = opt("tidynorm.silent"),
  .drop_orig = FALSE
)
```

Arguments

.data	A data frame of formant DCT coefficients
...	<tidy-select> One or more unquoted expressions separated by commas. These should target the vowel formant data columns.
.token_id_col	<data-masking> A column that identifies token ids.
.by	<tidy-select> A selection of columns to group by. Typically a column of speaker IDs.
.agg_by	<tidy-select> A selection of columns to to pre-aggregate by. Typically a column of vowel category labels.
.agg_fun	A function to use to pre-aggregate values.
.param_col	A column identifying the DCT parameter number.
.names	A glue::glue() expression for naming the normalized data columns. The "{.formant}" portion corresponds to the name of the original formant columns.
.silent	Suppress normalization information messages when running a norm_*() function. (Defaults to FALSE, overwritable using option 'tidynorm.silent' or environment variable 'R_TIDYNORM_TIDYNORM_SILENT')
.drop_orig	Should the originally targeted columns be dropped.

Details

$$\hat{F}_{ij} = \frac{F_{ij} - L_i}{S_i}$$

$$L_i = \frac{1}{N}\sum_{j=1}^{N} F_{ij}$$

$$S_i = \sqrt{\frac{\sum (F_{ij} - L_i)^2}{N - 1}}$$

Where

- $\hat{F}$ is the normalized formant
- i is the formant number
- j is the token number

Pre-aggregation:

To address imbalanced data across vowel classes, you can pre-aggregate the zeroth DCT coefficients by passing vowel class column(s) to .agg_by, and and an aggregating function to .agg_fun. The aggregation will be done after the DCT transform is applied, and will be done either formant intrinsically or formant extrinsically based on the setting passed to .by_formant.

Value

A data frame of Lobanov normalized DCT Coefficients.

References

Lobanov, B. (1971). Classification of Russian vowels spoken by different listeners. Journal of the Acoustical Society of America, 49, 606–608.

Examples

```
library(tidynorm)
library(dplyr)
ggplot2_inst <- require(ggplot2)

speaker_dct <- speaker_tracks |>
  reframe_with_dct(
    F1:F3,
    .by = speaker,
    .token_id_col = id,
    .time_col = t
  )

# Normalize DCT coefficients
speaker_dct_norm <- speaker_dct |>
  norm_dct_lobanov(
    F1:F3,
    .by = speaker,
    .token_id_col = id,
    .param_col = .param
  )

# Apply average and apply inverse dct
# to plot tracks
track_norm_means <- speaker_dct_norm |>
  summarise(
    .by = c(speaker, vowel, .param),
    across(
      ends_with("_z"),
      mean
    )
  ) |>
  reframe_with_idct(
    ends_with("_z"),
    .by = speaker,
    .token_id_col = vowel,
    .param_col = .param
  )


if (ggplot2_inst) {
  track_norm_means |>
    ggplot(
```

```
      aes(F2_z, F1_z, color = speaker)
    ) +
    geom_path(
      aes(
        group = interaction(speaker, vowel)
      )
    ) +
    scale_x_reverse() +
    scale_y_reverse() +
    scale_color_brewer(palette = "Dark2") +
    coord_fixed()
}
```

`norm_dct_nearey` *Nearey DCT Normalization*

Description

Nearey DCT Normalization

Usage

```
norm_dct_nearey(
  .data,
  ...,
  .token_id_col,
  .by = NULL,
  .by_formant = FALSE,
  .agg_by = NULL,
  .agg_fun = function(x) mean(x, na.rm = T),
  .param_col = NULL,
  .drop_orig = FALSE,
  .names = "{.formant}_lm",
  .silent = opt("tidynorm.silent")
)
```

Arguments

`.data`	A data frame containing vowel formant data
`...`	<tidy-select> One or more unquoted expressions separated by commas. These should target the vowel formant data columns.
`.token_id_col`	<data-masking> A column that identifies token ids.
`.by`	<tidy-select> A selection of columns to group by. Typically a column of speaker IDs.
`.by_formant`	Whether or not the normalization method is formant intrinsic.
`.agg_by`	<tidy-select> A selection of columns to to pre-aggregate by. Typically a column of vowel category labels.

`.agg_fun`	A function to use to pre-aggregate values.
`.param_col`	A column identifying the DCT parameter number.
`.drop_orig`	Should the originally targeted columns be dropped.
`.names`	A `glue::glue()` expression for naming the normalized data columns. The `"{.formant}"` portion corresponds to the name of the original formant columns.
`.silent`	Suppress normalization information messages when running a `norm_*()` function. (Defaults to `FALSE`, overwritable using option 'tidynorm.silent' or environment variable 'R_TIDYNORM_TIDYNORM_SILENT')

Details

Important: This function assumes that the DCT coefficients were estimated over log-transformed formant values.

When formant extrinsic:

$$\hat{F}_{ij} = \log(F_{ij}) - L$$

$$L = \frac{1}{MN} \sum_{i=1}^{M} \sum_{j=1}^{N} \log(F_{ij})$$

When formant intrinsic:

$$\hat{F}_{ij} = \log(F_{ij}) - L_i$$

$$L_i = \frac{1}{N} \sum_{j=1}^{N} \log(F_{ij})$$

Where

- $\hat{F}$ is the normalized formant
- i is the formant number
- j is the token number

Pre-aggregation:

To address imbalanced data across vowel classes, you can pre-aggregate the zeroth DCT coefficients by passing vowel class column(s) to `.agg_by`, and and an aggregating function to `.agg_fun`. The aggregation will be done after the DCT transform is applied, and will be done either formant intrinsically or formant extrinsically based on the setting passed to `.by_formant`.

Value

A data frame of Nearey normalized DCT coefficients

References

Nearey, T. M. (1978). Phonetic Feature Systems for Vowels [Ph.D.]. University of Alberta.

Examples

```
library(tidynorm)
library(dplyr)
ggplot2_inst <- require(ggplot2)

speaker_dct <- speaker_tracks |>
  mutate(
    across(
      F1:F3,
      log
    )
  ) |>
  reframe_with_dct(
    F1:F3,
    .by = speaker,
    .token_id_col = id,
    .time_col = t
  )

# Normalize DCT coefficients
speaker_dct_norm <- speaker_dct |>
  norm_dct_nearey(
    F1:F3,
    .by = speaker,
    .token_id_col = id,
    .param_col = .param
  )

# Apply average and apply inverse dct
# to plot tracks
track_norm_means <- speaker_dct_norm |>
  summarise(
    .by = c(speaker, vowel, .param),
    across(
      ends_with("_lm"),
      mean
    )
  ) |>
  reframe_with_idct(
    ends_with("_lm"),
    .by = speaker,
    .token_id_col = vowel,
    .param_col = .param
  )


if (ggplot2_inst) {
  track_norm_means |>
    ggplot(
      aes(F2_lm, F1_lm, color = speaker)
    ) +
    geom_path(
```

```
      aes(
        group = interaction(speaker, vowel)
      )
    ) +
    scale_x_reverse() +
    scale_y_reverse() +
    scale_color_brewer(palette = "Dark2") +
    coord_fixed()
}
```

`norm_dct_wattfab` *Watt and Fabricius DCT normalization*

Description

Watt and Fabricius DCT normalization

Usage

```
norm_dct_wattfab(
  .data,
  ...,
  .token_id_col,
  .by = NULL,
  .agg_by = NULL,
  .agg_fun = function(x) mean(x, na.rm = T),
  .param_col = NULL,
  .drop_orig = FALSE,
  .names = "{.formant}_wf",
  .silent = opt("tidynorm.silent")
)
```

Arguments

`.data`	A data frame containing vowel formant data
`...`	<tidy-select> One or more unquoted expressions separated by commas. These should target the vowel formant data columns.
`.token_id_col`	<data-masking> A column that identifies token ids.
`.by`	<tidy-select> A selection of columns to group by. Typically a column of speaker IDs.
`.agg_by`	<tidy-select> A selection of columns to to pre-aggregate by. Typically a column of vowel category labels.
`.agg_fun`	A function to use to pre-aggregate values.
`.param_col`	A column identifying the DCT parameter number.
`.drop_orig`	Should the originally targeted columns be dropped.

`.names`	A `glue::glue()` expression for naming the normalized data columns. The `"{.formant}"` portion corresponds to the name of the original formant columns.
`.silent`	Suppress normalization information messages when running a `norm_*()` function. (Defaults to `FALSE`, overwritable using option 'tidynorm.silent' or environment variable 'R_TIDYNORM_TIDYNORM_SILENT')

Details

This is a modified version of the Watt & Fabricius Method. The original method identified point vowels over which F1 and F2 centroids were calculated. The procedure here just identifies centroids by taking the mean of all formant values.

$$\hat{F}_{ij} = \frac{F_{ij}}{S_i}$$

$$S_i = \frac{1}{N}\sum_{j=1}^{N} F_{ij}$$

Where

- $\hat{F}$ is the normalized formant
- i is the formant number
- j is the token number

Pre-aggregation:

To address imbalanced data across vowel classes, you can pre-aggregate the zeroth DCT coefficients by passing vowel class column(s) to `.agg_by`, and and an aggregating function to `.agg_fun`. The aggregation will be done after the DCT transform is applied, and will be done either formant intrinsically or formant extrinsically based on the setting passed to `.by_formant`.

Value

A data frame of Watt & Fabricius normalized DCT coefficients.

References

Watt, D., & Fabricius, A. (2002). Evaluation of a technique for improving the mapping of multiple speakers' vowel spaces in the F1 ~ F2 plane. Leeds Working Papers in Linguistics and Phonetics, 9, 159–173.

Examples

```
library(tidynorm)
library(dplyr)
ggplot2_inst <- require(ggplot2)

speaker_dct <- speaker_tracks |>
  reframe_with_dct(
    F1:F3,
```

```
    .by = speaker,
    .token_id_col = id,
    .time_col = t
  )

# Normalize DCT coefficients
speaker_dct_norm <- speaker_dct |>
  norm_dct_wattfab(
    F1:F3,
    .by = speaker,
    .token_id_col = id,
    .param_col = .param
  )

# Apply average and apply inverse dct
# to plot tracks
track_norm_means <- speaker_dct_norm |>
  summarise(
    .by = c(speaker, vowel, .param),
    across(
      ends_with("_wf"),
      mean
    )
  ) |>
  reframe_with_idct(
    ends_with("_wf"),
    .by = speaker,
    .token_id_col = vowel,
    .param_col = .param
  )


if (ggplot2_inst) {
  track_norm_means |>
    ggplot(
      aes(F2_wf, F1_wf, color = speaker)
    ) +
    geom_path(
      aes(
        group = interaction(speaker, vowel)
      )
    ) +
    scale_x_reverse() +
    scale_y_reverse() +
    scale_color_brewer(palette = "Dark2") +
    coord_fixed()
}
```

---

norm_deltaF *Delta F Normalize*

Description

Delta F Normalize

Usage

```
norm_deltaF(
  .data,
  ...,
  .by = NULL,
  .agg_fun = function(x) mean(x, na.rm = T),
  .by_formant = FALSE,
  .agg_by = NULL,
  .drop_orig = FALSE,
  .keep_params = FALSE,
  .names = "{.formant}_df",
  .silent = opt("tidynorm.silent")
)
```

Arguments

.data	A data frame containing vowel formant data
...	<tidy-select> One or more unquoted expressions separated by commas. These should target the vowel formant data columns.
.by	<tidy-select> A selection of columns to group by. Typically a column of speaker IDs.
.agg_fun	A function to use to pre-aggregate values.
.by_formant	Ignored by this procedure
.agg_by	<tidy-select> A selection of columns to to pre-aggregate by. Typically a column of vowel category labels.
.drop_orig	Whether or not to drop the original formant data columns.
.keep_params	Whether or not to keep the Location (*_.L) and Scale (*_.S) normalization parameters
.names	A glue::glue() expression for naming the normalized data columns. The "{.formant}" portion corresponds to the name of the original formant columns.
.silent	Suppress normalization information messages when running a norm_*() function. (Defaults to FALSE, overwritable using option 'tidynorm.silent' or environment variable 'R_TIDYNORM_TIDYNORM_SILENT')

Details

$$\hat{F}_{ij} = \frac{F_{ij}}{S}$$

$$S = \frac{1}{MN} \sum_{i=1}^{M} \sum_{j=1}^{N} \frac{F_{ij}}{i - 0.5}$$

Where

- $\hat{F}$ is the normalized formant
- i is the formant number
- j is the token number

Pre-aggregation:

To address imbalanced data across vowel classes, you can pre-aggregate formant values by passing vowel class column(s) to `.agg_by`, and and an aggregating function to `.agg_fun`. The aggregation will be done after any `.pre_trans` transformation is applied, and will be done either formant intrinsically or formant extrinsically based on the setting passed to `.by_formant`.

Value

A data frame of Delta F normalized formant values.

References

Johnson, K. (2020). The ΔF method of vocal tract length normalization for vowels. Laboratory Phonology: Journal of the Association for Laboratory Phonology, 11(1), Article 1. doi:10.5334/labphon.196

Examples

```
library(tidynorm)
ggplot2_inst <- require(ggplot2)

speaker_data_deltaF <- speaker_data |>
  norm_deltaF(
    F1:F3,
    .by = speaker,
    .names = "{.formant}_df"
  )

if (ggplot2_inst) {
  ggplot(
    speaker_data_deltaF,
    aes(
      F2_df,
      F1_df,
      color = speaker
    )
  ) +
    stat_density_2d(
      bins = 4
    ) +
    scale_color_brewer(
      palette = "Dark2"
    ) +
    scale_x_reverse() +
    scale_y_reverse() +
    coord_fixed()
}
```

norm_generic *Generic Normalization Procedure*

Description

This is a generic normalization procedure with which you can create your own normalization method.

Usage

```
norm_generic(
  .data,
  ...,
  .by = NULL,
  .by_formant = FALSE,
  .by_token = FALSE,
  .agg_by = NULL,
  .agg_fun = function(x) mean(x, na.rm = T),
  .L = 0,
  .S = 1,
  .pre_trans = identity,
  .post_trans = identity,
  .drop_orig = FALSE,
  .keep_params = FALSE,
  .names = "{.formant}_n",
  .silent = opt("tidynorm.silent"),
  .call = caller_env()
)
```

Arguments

.data	A data frame containing vowel formant data
...	<tidy-select> One or more unquoted expressions separated by commas. These should target the vowel formant data columns.
.by	<tidy-select> A selection of columns to group by. Typically a column of speaker IDs.
.by_formant	Whether or not the normalization method is formant intrinsic.
.by_token	Whether or not the normalization method is vowel intrinsic
.agg_by	<tidy-select> A selection of columns to to pre-aggregate by. Typically a column of vowel category labels.
.agg_fun	A function to use to pre-aggregate values.
.L	An expression defining the location parameter. See Details for more information.
.S	An expression defining the scale parameter. See Details for more information.

.pre_trans	A function to apply to formant values before normalization.
.post_trans	A function to apply to formant values after normalization.
.drop_orig	Whether or not to drop the original formant data columns.
.keep_params	Whether or not to keep the Location (*_.L) and Scale (*_.S) normalization parameters
.names	A glue::glue() expression for naming the normalized data columns. The "{.formant}" portion corresponds to the name of the original formant columns.
.silent	Suppress normalization information messages when running a norm_*() function. (Defaults to FALSE, overwritable using option 'tidynorm.silent' or environment variable 'R_TIDYNORM_TIDYNORM_SILENT')
.call	Used for internal purposes.

Details

The following norm_* procedures are built on top of norm_generic().

- norm_lobanov
- norm_nearey
- norm_deltaF
- norm_wattfab
- norm_barkz

Location and Scale expressions:

All normalization procedures built on norm_generic produce normalized formant values ($\hat{F}$) by subtracting a location parameter (L) and dividing by a scale parameter (S).

$$\hat{F} = \frac{F - L}{S}$$

The expressions for calculating L and S can be passed to .L and .S, respectively. Available values for these expressions are

.formant The original formant value

.formant_num The number of the formant. (e.g. 1 for F1, 2 for F2 etc)

Along with any data columns from your original data.

Pre and Post normalization transforms:

To apply any transformations before or after normalization, you can pass a function to .pre_trans and .post_trans.

Formant In/Extrinsic Normalization:

If .by_formant is TRUE, normalization will be formant intrinsic. If .by_formant is FALSE, normalization will be formant extrinsic.

Token In/Extrinsic Normalization:

If .by_token is TRUE, normalization will be token intrinsic. If .by_token is FALSE, normalization will be token extrinsic.

Pre-aggregation:

To address imbalanced data across vowel classes, you can pre-aggregate formant values by passing vowel class column(s) to `.agg_by`, and and an aggregating function to `.agg_fun`. The aggregation will be done after any `.pre_trans` transformation is applied, and will be done either formant intrinsically or formant extrinsically based on the setting passed to `.by_formant`.

Value

A data frame of normalized formant values

Examples

```
library(tidynorm)
library(dplyr)

speaker_data |>
  norm_generic(
    F1:F3,
    .by = speaker,
    .by_formant = TRUE,
    .L = median(.formant, na.rm = TRUE),
    .S = mad(.formant, na.rm = TRUE),
    .drop_orig = TRUE,
    .names = "{.formant}_mad"
  )
```

---

norm_lobanov *Lobanov Normalize*

Description

Lobanov Normalize

Usage

```
norm_lobanov(
  .data,
  ...,
  .by = NULL,
  .by_formant = TRUE,
  .agg_by = NULL,
  .agg_fun = function(x) mean(x, na.rm = T),
  .drop_orig = FALSE,
  .keep_params = FALSE,
  .names = "{.formant}_z",
  .silent = opt("tidynorm.silent")
)
```

Arguments

.data	A data frame containing vowel formant data
...	<tidy-select> One or more unquoted expressions separated by commas. These should target the vowel formant data columns.
.by	<tidy-select> A selection of columns to group by. Typically a column of speaker IDs.
.by_formant	Ignored by this procedure
.agg_by	<tidy-select> A selection of columns to to pre-aggregate by. Typically a column of vowel category labels.
.agg_fun	A function to use to pre-aggregate values.
.drop_orig	Whether or not to drop the original formant data columns.
.keep_params	Whether or not to keep the Location (*_.L) and Scale (*_.S) normalization parameters
.names	A glue::glue() expression for naming the normalized data columns. The "{.formant}" portion corresponds to the name of the original formant columns.
.silent	Suppress normalization information messages when running a norm_*() function. (Defaults to FALSE, overwritable using option 'tidynorm.silent' or environment variable 'R_TIDYNORM_TIDYNORM_SILENT')

Details

$$\hat{F}_{ij} = \frac{F_{ij} - L_i}{S_i}$$

$$L_i = \frac{1}{N}\sum_{j=1}^{N} F_{ij}$$

$$S_i = \sqrt{\frac{\sum (F_{ij} - L_i)^2}{N - 1}}$$

Where

- $\hat{F}$ is the normalized formant
- i is the formant number
- j is the token number

Pre-aggregation:

To address imbalanced data across vowel classes (Lobanov 2.0 (Brand et al., 2021)), you can pre-aggregate formant values by passing vowel class column(s) to .agg_by, and and an aggregating function to .agg_fun. The aggregation will be done after any .pre_trans transformation is applied, and will be done either formant intrinsically or formant extrinsically based on the setting passed to .by_formant.

Value

A data frame of Lobanov normalized formant values.

References

Lobanov, B. (1971). Classification of Russian vowels spoken by different listeners. Journal of the Acoustical Society of America, 49, 606–608.

Brand, J., Hay, J., Clark, L., Watson, K., & Sóskuthy, M. (2021). Systematic co-variation of monophthongs across speakers of New Zealand English. Journal of Phonetics, 88, 101096. doi:10.1016/j.wocn.2021.101096

Examples

```
library(tidynorm)
ggplot2_inst <- require(ggplot2)

speaker_data_lobanov <- speaker_data |>
  norm_lobanov(
    F1:F3,
    .by = speaker,
    .names = "{.formant}_z"
  )

if (ggplot2_inst) {
  ggplot(
    speaker_data_lobanov,
    aes(
      F2_z,
      F1_z,
      color = speaker
    )
  ) +
    stat_density_2d(
      bins = 4
    ) +
    scale_color_brewer(
      palette = "Dark2"
    ) +
    scale_x_reverse() +
    scale_y_reverse() +
    coord_fixed()
}
```

---

norm_nearey *Nearey Normalize*

Description

Nearey Normalize

Usage

```
norm_nearey(
  .data,
  ...,
  .by = NULL,
  .by_formant = FALSE,
  .agg_by = NULL,
  .agg_fun = function(x) mean(x, na.rm = T),
  .drop_orig = FALSE,
  .keep_params = FALSE,
  .names = "{.formant}_lm",
  .silent = opt("tidynorm.silent")
)
```

Arguments

.data	A data frame containing vowel formant data
...	<tidy-select> One or more unquoted expressions separated by commas. These should target the vowel formant data columns.
.by	<tidy-select> A selection of columns to group by. Typically a column of speaker IDs.
.by_formant	Whether or not the normalization method is formant intrinsic.
.agg_by	<tidy-select> A selection of columns to to pre-aggregate by. Typically a column of vowel category labels.
.agg_fun	A function to use to pre-aggregate values.
.drop_orig	Whether or not to drop the original formant data columns.
.keep_params	Whether or not to keep the Location (*_.L) and Scale (*_.S) normalization parameters
.names	A glue::glue() expression for naming the normalized data columns. The "{.formant}" portion corresponds to the name of the original formant columns.
.silent	Suppress normalization information messages when running a norm_*() function. (Defaults to FALSE, overwritable using option 'tidynorm.silent' or environment variable 'R_TIDYNORM_TIDYNORM_SILENT')

Details

When formant extrinsic:

$$\hat{F}_{ij} = \log(F_{ij}) - L$$

$$L = \frac{1}{MN} \sum_{i=1}^{M} \sum_{j=1}^{N} \log(F_{ij})$$

When formant intrinsic:

$$\hat{F}_{ij} = \log(F_{ij}) - L_i$$

$$L_i = \frac{1}{N}\sum_{j=1}^{N}\log(F_{ij})$$

Where

- $\hat{F}$ is the normalized formant
- i is the formant number
- j is the token number

Pre-aggregation:

To address imbalanced data across vowel classes, you can pre-aggregate formant values by passing vowel class column(s) to `.agg_by`, and and an aggregating function to `.agg_fun`. The aggregation will be done after any `.pre_trans` transformation is applied, and will be done either formant intrinsically or formant extrinsically based on the setting passed to `.by_formant`.

Value

A data frame of Nearey normalized formant values.

References

Nearey, T. M. (1978). Phonetic Feature Systems for Vowels [Ph.D.]. University of Alberta.

Examples

```
library(tidynorm)
ggplot2_inst <- require(ggplot2)

speaker_data_nearey <- speaker_data |>
  norm_nearey(
    F1:F3,
    .by = speaker,
    .by_formant = FALSE,
    .names = "{.formant}_nearey"
  )

if (ggplot2_inst) {
  ggplot(
    speaker_data_nearey,
    aes(
      F2_nearey,
      F1_nearey,
      color = speaker
    )
  ) +
    stat_density_2d(
      bins = 4
    ) +
    scale_color_brewer(
      palette = "Dark2"
```

```
    ) +
    scale_x_reverse() +
    scale_y_reverse() +
    coord_fixed() +
    labs(
      title = "Formant extrinsic"
    )
}

speaker_data_nearey2 <- speaker_data |>
  norm_nearey(
    F1:F3,
    .by = speaker,
    .by_formant = TRUE,
    .names = "{.formant}_nearey"
  )

if (ggplot2_inst) {
  ggplot(
    speaker_data_nearey2,
    aes(
      F2_nearey,
      F1_nearey,
      color = speaker
    )
  ) +
    stat_density_2d(
      bins = 4
    ) +
    scale_color_brewer(
      palette = "Dark2"
    ) +
    scale_x_reverse() +
    scale_y_reverse() +
    coord_fixed() +
    labs(
      title = "Formant intrinsic"
    )
}
```

`norm_track_barkz` *Bark Difference Track Normalization*

Description

Bark Difference Track Normalization

Usage

```
norm_track_barkz(
```

```
  .data,
  ...,
  .token_id_col,
  .by = NULL,
  .time_col = NULL,
  .order = 5,
  .return_dct = FALSE,
  .drop_orig = FALSE,
  .names = "{.formant}_bz",
  .silent = opt("tidynorm.silent")
)
```

Arguments

.data	A data frame containing vowel formant data
...	<tidy-select> One or more unquoted expressions separated by commas. These should target the vowel formant data columns.
.token_id_col	<data-masking> A column that identifies token ids.
.by	<tidy-select> A selection of columns to group by. Typically a column of speaker IDs.
.time_col	<data-masking> A time column. (optional)
.order	The number of DCT parameters to use.
.return_dct	Whether or not the normalized DCT coefficients themselves should be returned.
.drop_orig	Should the originally targeted columns be dropped.
.names	A glue::glue() expression for naming the normalized data columns. The "{.formant}" portion corresponds to the name of the original formant columns.
.silent	Suppress normalization information messages when running a norm_*() function. (Defaults to FALSE, overwritable using option 'tidynorm.silent' or environment variable 'R_TIDYNORM_TIDYNORM_SILENT')

Details

This is a within-token normalization technique. First all formants are converted to Bark (see hz_to_bark), then, within each token, F3 is subtracted from F1 and F2.

$$\hat{F}_{ij} = F_{ij} - L_j$$

$$L_j = F_{3j}$$

Value

A data frame of either normalized formant tracks, or normalized DCT parameters.

A data frame of Bark difference normalized formant tracks.

References

Syrdal, A. K., & Gopal, H. S. (1986). A perceptual model of vowel recognition based on the auditory representation of American English vowels. The Journal of the Acoustical Society of America, 79(4), 1086–1100. doi:10.1121/1.393381

Examples

```
library(tidynorm)
library(dplyr)
ggplot2_inst <- require(ggplot2)

track_subset <- speaker_tracks |>
  filter(
    .by = c(speaker, id),
    if_all(
      F1:F3,
      .fns = \(x) mean(is.finite(x)) > 0.9
    ),
    row_number() %% 2 == 1
  )

track_norm <- track_subset |>
  norm_track_barkz(
    F1:F3,
    .by = speaker,
    .token_id_col = id,
    .time_col = t,
    .drop_orig = TRUE
  )

if (ggplot2_inst) {
  track_norm |>
    ggplot(
      aes(F2_bz, F1_bz, color = speaker)
    ) +
    stat_density_2d(bins = 4) +
    scale_x_reverse() +
    scale_y_reverse() +
    scale_color_brewer(palette = "Dark2") +
    coord_fixed()
}


# returning the DCT coefficients
track_norm_dct <- track_subset |>
  norm_track_barkz(
    F1:F3,
    .by = speaker,
    .token_id_col = id,
    .time_col = t,
    .drop_orig = TRUE,
    .return_dct = TRUE,
```

```
    .names = "{.formant}_bz"
  )

track_norm_means <- track_norm_dct |>
  summarise(
    .by = c(speaker, vowel, .param),
    across(
      ends_with("_bz"),
      mean
    )
  ) |>
  reframe_with_idct(
    ends_with("_bz"),
    .by = speaker,
    .token_id_col = vowel,
    .param_col = .param
  )


if (ggplot2_inst) {
  track_norm_means |>
    ggplot(
      aes(F2_bz, F1_bz, color = speaker)
    ) +
    geom_path(
      aes(
        group = interaction(speaker, vowel)
      )
    ) +
    scale_x_reverse() +
    scale_y_reverse() +
    scale_color_brewer(palette = "Dark2") +
    coord_fixed()
}
```

`norm_track_deltaF` *Delta F Track Normalization*

Description

Delta F Track Normalization

Usage

```
norm_track_deltaF(
  .data,
  ...,
  .token_id_col,
  .by = NULL,
  .agg_by = NULL,
```

```
  .agg_fun = function(x) mean(x, na.rm = T),
  .time_col = NULL,
  .order = 5,
  .return_dct = FALSE,
  .drop_orig = FALSE,
  .names = "{.formant}_df",
  .silent = opt("tidynorm.silent")
)
```

Arguments

.data	A data frame containing vowel formant data
...	<tidy-select> One or more unquoted expressions separated by commas. These should target the vowel formant data columns.
.token_id_col	<data-masking> A column that identifies token ids.
.by	<tidy-select> A selection of columns to group by. Typically a column of speaker IDs.
.agg_by	<tidy-select> A selection of columns to to pre-aggregate by. Typically a column of vowel category labels.
.agg_fun	A function to use to pre-aggregate values.
.time_col	<data-masking> A time column. (optional)
.order	The number of DCT parameters to use.
.return_dct	Whether or not the normalized DCT coefficients themselves should be returned.
.drop_orig	Should the originally targeted columns be dropped.
.names	A glue::glue() expression for naming the normalized data columns. The "{.formant}" portion corresponds to the name of the original formant columns.
.silent	Suppress normalization information messages when running a norm_*() function. (Defaults to FALSE, overwritable using option 'tidynorm.silent' or environment variable 'R_TIDYNORM_TIDYNORM_SILENT')

Details

$$\hat{F}_{ij} = \frac{F_{ij}}{S}$$

$$S = \frac{1}{MN} \sum_{i=1}^{M} \sum_{j=1}^{N} \frac{F_{ij}}{i - 0.5}$$

Where

- $\hat{F}$ is the normalized formant
- i is the formant number
- j is the token number

Pre-aggregation:

To address imbalanced data across vowel classes, you can pre-aggregate the zeroth DCT coefficients by passing vowel class column(s) to `.agg_by`, and and an aggregating function to `.agg_fun`. The aggregation will be done after the DCT transform is applied, and will be done either formant intrinsically or formant extrinsically based on the setting passed to `.by_formant`.

Value

A data frame of Delta F normalized formant tracks.

References

Johnson, K. (2020). The ΔF method of vocal tract length normalization for vowels. Laboratory Phonology: Journal of the Association for Laboratory Phonology, 11(1), Article 1. doi:10.5334/labphon.196

Examples

```
library(tidynorm)
library(dplyr)
ggplot2_inst <- require(ggplot2)

track_subset <- speaker_tracks |>
  filter(
    .by = c(speaker, id),
    if_all(
      F1:F3,
      .fns = \(x) mean(is.finite(x)) > 0.9
    ),
    row_number() %% 2 == 1
  )

track_norm <- track_subset |>
  norm_track_deltaF(
    F1:F3,
    .by = speaker,
    .token_id_col = id,
    .time_col = t,
    .drop_orig = TRUE
  )

if (ggplot2_inst) {
  track_norm |>
    ggplot(
      aes(F2_df, F1_df, color = speaker)
    ) +
    stat_density_2d(bins = 4) +
    scale_x_reverse() +
    scale_y_reverse() +
    scale_color_brewer(palette = "Dark2") +
    coord_fixed()
}
```

```
# returning the DCT coefficients
track_norm_dct <- track_subset |>
  norm_track_deltaF(
    F1:F3,
    .by = speaker,
    .token_id_col = id,
    .time_col = t,
    .drop_orig = TRUE,
    .return_dct = TRUE,
    .names = "{.formant}_df"
  )

track_norm_means <- track_norm_dct |>
  summarise(
    .by = c(speaker, vowel, .param),
    across(
      ends_with("_df"),
      mean
    )
  ) |>
  reframe_with_idct(
    ends_with("_df"),
    .by = speaker,
    .token_id_col = vowel,
    .param_col = .param
  )


if (ggplot2_inst) {
  track_norm_means |>
    ggplot(
      aes(F2_df, F1_df, color = speaker)
    ) +
    geom_path(
      aes(
        group = interaction(speaker, vowel)
      )
    ) +
    scale_x_reverse() +
    scale_y_reverse() +
    scale_color_brewer(palette = "Dark2") +
    coord_fixed()
}
```

`norm_track_generic` *Generic Formant Track Normalization Procedure*

Description

Normalize formant tracks using Discrete Cosine Transform normalization

Usage

```
norm_track_generic(
  .data,
  ...,
  .token_id_col,
  .by = NULL,
  .by_formant = FALSE,
  .by_token = FALSE,
  .agg_by = NULL,
  .agg_fun = function(x) mean(x, na.rm = T),
  .time_col = NULL,
  .L = 0,
  .S = 1/sqrt(2),
  .pre_trans = identity,
  .post_trans = identity,
  .order = 5,
  .return_dct = FALSE,
  .drop_orig = FALSE,
  .names = "{.formant}_n",
  .silent = opt("tidynorm.silent"),
  .call = caller_env()
)
```

Arguments

.data	A data frame containing vowel formant data
...	<tidy-select> One or more unquoted expressions separated by commas. These should target the vowel formant data columns.
.token_id_col	<data-masking> A column that identifies token ids.
.by	<tidy-select> A selection of columns to group by. Typically a column of speaker IDs.
.by_formant	Whether or not the normalization method is formant intrinsic.
.by_token	Whether or not the normalization method is token intrinsic
.agg_by	<tidy-select> A selection of columns to to pre-aggregate by. Typically a column of vowel category labels.
.agg_fun	A function to use to pre-aggregate values.
.time_col	<data-masking> A time column. (optional)
.L	An expression defining the location parameter. See Details for more information.
.S	An expression defining the scale parameter. See Details for more information.
.pre_trans	A function to apply to formant values before normalization.
.post_trans	A function to apply to formant values after normalization.
.order	The number of DCT parameters to use.
.return_dct	Whether or not the normalized DCT coefficients themselves should be returned.

`.drop_orig`	Should the originally targeted columns be dropped.
`.names`	A `glue::glue()` expression for naming the normalized data columns. The `"{.formant}"` portion corresponds to the name of the original formant columns.
`.silent`	Suppress normalization information messages when running a `norm_*()` function. (Defaults to `FALSE`, overwritable using option 'tidynorm.silent' or environment variable 'R_TIDYNORM_TIDYNORM_SILENT')
`.call`	Used for internal purposes.

Details

The following `norm_track_*` procedures were built on top of `norm_track_generic`.

- norm_track_lobanov
- norm_track_nearey
- norm_track_deltaF
- norm_track_wattfab
- norm_track_barkz

This will normalize vowel formant tracks in the following steps:

1. Any `.pre_trans` transformations will be applied to the formant data.
2. The Discrete Cosine Transform will be applied to the formant data.
3. Location `.L` and Scale `.S` expressions will be used to summarize the zeroth DCT coefficients.
4. These location and scale parameters will be used to normalize the DCT coefficients.
5. If `.return_dct = TRUE`, these normalized DCT coefficients will be returned. Otherwise, the Inverse Discrete Cosine Transform will be applied to the normalized DCT coefficients.
6. Any `.post_trans` transformations will be applied.

Location and Scale expressions:

All normalization procedures built on norm_track_generic work by normalizing DCT coefficients directly. If F_k is the kth DCT coefficient the normalization procedure is

$$\hat{F}_k = \frac{F_k - L'}{\sqrt{2}S}$$

$$L' = \begin{cases} L & \text{for } k = 0 \\ 0 & \text{for } k > 0 \end{cases}$$

Rather than requiring users to remember to multiply expressions for S by $\sqrt{2}$, this is done by norm_track_generic itself, to allow greater parallelism with how norm_generic works.

Note: If you want to scale values by a constant in the normalization, you'll need to divide the constant by `sqrt(2)`. Post-normalization scaling (e.g. re-scaling to formant-like values) is probably best handled with a function passed to `.post_trans`.

The expressions for calculating L and S can be passed to `.L` and `.S`, respectively. Available values for these expressions are

`.formant` The original formant value

`.formant_num` The number of the formant. (e.g. 1 for F1, 2 for F2 etc)

Along with any data columns from your original data.

Identifying tokens:

Track normalization requires identifying individual tokens, so there must be a column that uniquely identifies (or, in combination with a `.by` grouping, uniquely identifies) each individual token. This column should be passed to `.token_id_col`.

Order:

The number of DCT coefficients used is defined by `.order`. The default value is 5. Larger numbers will lead to less smoothing, and smaller numbers will lead to more smoothing.

Pre-aggregation:

To address imbalanced data across vowel classes, you can pre-aggregate the zeroth DCT coefficients by passing vowel class column(s) to `.agg_by`, and and an aggregating function to `.agg_fun`. The aggregation will be done after the DCT transform is applied, and will be done either formant intrinsically or formant extrinsically based on the setting passed to `.by_formant`.

Value

A data frame of normalized formant tracks.

Examples

```
library(tidynorm)
library(dplyr)
ggplot2_inst <- require(ggplot2)

track_subset <- speaker_tracks |>
  filter(
    .by = c(speaker, id),
    if_all(
      F1:F3,
      .fns = \(x) mean(is.finite(x)) > 0.9
    ),
    row_number() %% 2 == 1
  )

track_norm <- track_subset |>
  norm_track_generic(
    F1:F3,
    .by = speaker,
    .token_id_col = id,
    .by_formant = TRUE,
    .L = median(.formant, na.rm = TRUE),
    .S = mad(.formant, na.rm = TRUE),
    .time_col = t,
    .drop_orig = TRUE,
    .names = "{.formant}_mad"
  )
```

```
if (ggplot2_inst) {
  track_norm |>
    ggplot(
      aes(F2_mad, F1_mad, color = speaker)
    ) +
    stat_density_2d(bins = 4) +
    scale_x_reverse() +
    scale_y_reverse() +
    scale_color_brewer(palette = "Dark2") +
    coord_fixed()
}

# returning the DCT coefficients
track_norm_dct <- track_subset |>
  norm_track_generic(
    F1:F3,
    .by = speaker,
    .token_id_col = id,
    .by_formant = TRUE,
    .L = median(.formant, na.rm = TRUE),
    .S = mad(.formant, na.rm = TRUE),
    .time_col = t,
    .drop_orig = TRUE,
    .return_dct = TRUE,
    .names = "{.formant}_mad"
  )

track_norm_means <- track_norm_dct |>
  summarise(
    .by = c(speaker, vowel, .param),
    across(
      ends_with("_mad"),
      mean
    )
  ) |>
  reframe_with_idct(
    ends_with("_mad"),
    .by = speaker,
    .token_id_col = vowel,
    .param_col = .param
  )


if (ggplot2_inst) {
  track_norm_means |>
    ggplot(
      aes(F2_mad, F1_mad, color = speaker)
    ) +
    geom_path(
      aes(
        group = interaction(speaker, vowel)
      )
    ) +
```

```
    scale_x_reverse() +
    scale_y_reverse() +
    scale_color_brewer(palette = "Dark2") +
    coord_fixed()
}
```

`norm_track_lobanov` *Lobanov Track Normalization*

Description

Lobanov Track Normalization

Usage

```
norm_track_lobanov(
  .data,
  ...,
  .token_id_col,
  .by = NULL,
  .agg_by = NULL,
  .agg_fun = function(x) mean(x, na.rm = T),
  .time_col = NULL,
  .order = 5,
  .return_dct = FALSE,
  .drop_orig = FALSE,
  .names = "{.formant}_z",
  .silent = opt("tidynorm.silent")
)
```

Arguments

`.data`	A data frame containing vowel formant data
`...`	<tidy-select> One or more unquoted expressions separated by commas. These should target the vowel formant data columns.
`.token_id_col`	<data-masking> A column that identifies token ids.
`.by`	<tidy-select> A selection of columns to group by. Typically a column of speaker IDs.
`.agg_by`	<tidy-select> A selection of columns to to pre-aggregate by. Typically a column of vowel category labels.
`.agg_fun`	A function to use to pre-aggregate values.
`.time_col`	<data-masking> A time column. (optional)
`.order`	The number of DCT parameters to use.
`.return_dct`	Whether or not the normalized DCT coefficients themselves should be returned.
`.drop_orig`	Should the originally targeted columns be dropped.

`.names`	A `glue::glue()` expression for naming the normalized data columns. The `"{.formant}"` portion corresponds to the name of the original formant columns.
`.silent`	Suppress normalization information messages when running a `norm_*()` function. (Defaults to `FALSE`, overwritable using option 'tidynorm.silent' or environment variable 'R_TIDYNORM_TIDYNORM_SILENT')

Details

$$\hat{F}_{ij} = \frac{F_{ij} - L_i}{S_i}$$

$$L_i = \frac{1}{N}\sum_{j=1}^{N} F_{ij}$$

$$S_i = \sqrt{\frac{\sum (F_{ij} - L_i)^2}{N - 1}}$$

Where

- $\hat{F}$ is the normalized formant
- i is the formant number
- j is the token number

Pre-aggregation:

To address imbalanced data across vowel classes, you can pre-aggregate the zeroth DCT coefficients by passing vowel class column(s) to `.agg_by`, and and an aggregating function to `.agg_fun`. The aggregation will be done after the DCT transform is applied, and will be done either formant intrinsically or formant extrinsically based on the setting passed to `.by_formant`.

Value

A data frame of Lobanov normalized formant tracks.

References

Lobanov, B. (1971). Classification of Russian vowels spoken by different listeners. Journal of the Acoustical Society of America, 49, 606–608.

Examples

```
library(tidynorm)
library(dplyr)
ggplot2_inst <- require(ggplot2)

track_subset <- speaker_tracks |>
  filter(
    .by = c(speaker, id),
    if_all(
```

```
      F1:F3,
      .fns = \(x) mean(is.finite(x)) > 0.9
    ),
    row_number() %% 2 == 1
  )

track_norm <- track_subset |>
  norm_track_lobanov(
    F1:F3,
    .by = speaker,
    .token_id_col = id,
    .time_col = t,
    .drop_orig = TRUE
  )

if (ggplot2_inst) {
  track_norm |>
    ggplot(
      aes(F2_z, F1_z, color = speaker)
    ) +
    stat_density_2d(bins = 4) +
    scale_x_reverse() +
    scale_y_reverse() +
    scale_color_brewer(palette = "Dark2") +
    coord_fixed()
}

# returning the DCT coefficients
track_norm_dct <- track_subset |>
  norm_track_lobanov(
    F1:F3,
    .by = speaker,
    .token_id_col = id,
    .time_col = t,
    .return_dct = TRUE,
    .drop_orig = TRUE,
    .names = "{.formant}_z"
  )

track_norm_means <- track_norm_dct |>
  summarise(
    .by = c(speaker, vowel, .param),
    across(
      ends_with("_z"),
      mean
    )
  ) |>
  reframe_with_idct(
    ends_with("_z"),
    .by = speaker,
    .token_id_col = vowel,
    .param_col = .param
  )
```

```
if (ggplot2_inst) {
  track_norm_means |>
    ggplot(
      aes(F2_z, F1_z, color = speaker)
    ) +
    geom_path(
      aes(
        group = interaction(speaker, vowel)
      )
    ) +
    scale_x_reverse() +
    scale_y_reverse() +
    scale_color_brewer(palette = "Dark2") +
    coord_fixed()
}
```

`norm_track_nearey` *Nearey Track Normalization*

Description

Nearey Track Normalization

Usage

```
norm_track_nearey(
  .data,
  ...,
  .token_id_col,
  .by = NULL,
  .by_formant = FALSE,
  .agg_by = NULL,
  .agg_fun = function(x) mean(x, na.rm = T),
  .time_col = NULL,
  .order = 5,
  .return_dct = FALSE,
  .drop_orig = FALSE,
  .names = "{.formant}_lm",
  .silent = opt("tidynorm.silent")
)
```

Arguments

`.data`	A data frame containing vowel formant data
`...`	<tidy-select> One or more unquoted expressions separated by commas. These should target the vowel formant data columns.

.token_id_col	<data-masking> A column that identifies token ids.
.by	<tidy-select> A selection of columns to group by. Typically a column of speaker IDs.
.by_formant	Whether or not the normalization method is formant intrinsic.
.agg_by	<tidy-select> A selection of columns to to pre-aggregate by. Typically a column of vowel category labels.
.agg_fun	A function to use to pre-aggregate values.
.time_col	<data-masking> A time column. (optional)
.order	The number of DCT parameters to use.
.return_dct	Whether or not the normalized DCT coefficients themselves should be returned.
.drop_orig	Should the originally targeted columns be dropped.
.names	A glue::glue() expression for naming the normalized data columns. The "{.formant}" portion corresponds to the name of the original formant columns.
.silent	Suppress normalization information messages when running a norm_*() function. (Defaults to FALSE, overwritable using option 'tidynorm.silent' or environment variable 'R_TIDYNORM_TIDYNORM_SILENT')

Details

When formant extrinsic:

$$\hat{F}_{ij} = \log(F_{ij}) - L$$

$$L = \frac{1}{MN} \sum_{i=1}^{M} \sum_{j=1}^{N} \log(F_{ij})$$

When formant intrinsic:

$$\hat{F}_{ij} = \log(F_{ij}) - L_i$$

$$L_i = \frac{1}{N} \sum_{j=1}^{N} \log(F_{ij})$$

Where

- $\hat{F}$ is the normalized formant
- i is the formant number
- j is the token number

Pre-aggregation:

To address imbalanced data across vowel classes, you can pre-aggregate the zeroth DCT coefficients by passing vowel class column(s) to .agg_by, and and an aggregating function to .agg_fun. The aggregation will be done after the DCT transform is applied, and will be done either formant intrinsically or formant extrinsically based on the setting passed to .by_formant.

Value

A data frame of Nearey normalized formant tracks.

References

Nearey, T. M. (1978). Phonetic Feature Systems for Vowels [Ph.D.]. University of Alberta.

Examples

```
library(tidynorm)
library(dplyr)
ggplot2_inst <- require(ggplot2)

track_subset <- speaker_tracks |>
  filter(
    .by = c(speaker, id),
    if_all(
      F1:F3,
      .fns = \(x) mean(is.finite(x)) > 0.9
    ),
    row_number() %% 2 == 1
  )

track_norm <- track_subset |>
  norm_track_nearey(
    F1:F3,
    .by = speaker,
    .token_id_col = id,
    .time_col = t,
    .by_formant = TRUE,
    .drop_orig = TRUE
  )

if (ggplot2_inst) {
  track_norm |>
    ggplot(
      aes(F2_lm, F1_lm, color = speaker)
    ) +
    stat_density_2d(bins = 4) +
    scale_x_reverse() +
    scale_y_reverse() +
    scale_color_brewer(palette = "Dark2") +
    coord_fixed()
}


# returning the DCT coefficients
track_norm_dct <- track_subset |>
  norm_track_nearey(
    F1:F3,
    .by = speaker,
    .token_id_col = id,
    .time_col = t,
    .by_formant = FALSE,
    .drop_orig = TRUE,
    .return_dct = TRUE,
```

```
    .names = "{.formant}_lm"
  )

track_norm_means <- track_norm_dct |>
  summarise(
    .by = c(speaker, vowel, .param),
    across(
      ends_with("_lm"),
      mean
    )
  ) |>
  reframe_with_idct(
    ends_with("_lm"),
    .by = speaker,
    .token_id_col = vowel,
    .param_col = .param
  )


if (ggplot2_inst) {
  track_norm_means |>
    ggplot(
      aes(F2_lm, F1_lm, color = speaker)
    ) +
    geom_path(
      aes(
        group = interaction(speaker, vowel)
      )
    ) +
    scale_x_reverse() +
    scale_y_reverse() +
    scale_color_brewer(palette = "Dark2") +
    coord_fixed()
}
```

`norm_track_wattfab` *Watt and Fabricius Track normalization*

Description

Watt and Fabricius Track normalization

Usage

```
norm_track_wattfab(
  .data,
  ...,
  .token_id_col,
  .by = NULL,
  .agg_by = NULL,
```

```
  .agg_fun = function(x) mean(x, na.rm = T),
  .time_col = NULL,
  .order = 5,
  .return_dct = FALSE,
  .drop_orig = FALSE,
  .names = "{.formant}_wf",
  .silent = opt("tidynorm.silent")
)
```

Arguments

.data	A data frame containing vowel formant data
...	<tidy-select> One or more unquoted expressions separated by commas. These should target the vowel formant data columns.
.token_id_col	<data-masking> A column that identifies token ids.
.by	<tidy-select> A selection of columns to group by. Typically a column of speaker IDs.
.agg_by	<tidy-select> A selection of columns to to pre-aggregate by. Typically a column of vowel category labels.
.agg_fun	A function to use to pre-aggregate values.
.time_col	<data-masking> A time column. (optional)
.order	The number of DCT parameters to use.
.return_dct	Whether or not the normalized DCT coefficients themselves should be returned.
.drop_orig	Should the originally targeted columns be dropped.
.names	A glue::glue() expression for naming the normalized data columns. The "{.formant}" portion corresponds to the name of the original formant columns.
.silent	Suppress normalization information messages when running a norm_*() function. (Defaults to FALSE, overwritable using option 'tidynorm.silent' or environment variable 'R_TIDYNORM_TIDYNORM_SILENT')

Details

This is a modified version of the Watt & Fabricius Method. The original method identified point vowels over which F1 and F2 centroids were calculated. The procedure here just identifies centroids by taking the mean of all formant values.

$$\hat{F}_{ij} = \frac{F_{ij}}{S_i}$$

$$S_i = \frac{1}{N}\sum_{j=1}^{N} F_{ij}$$

Where

- $\hat{F}$ is the normalized formant
- i is the formant number
- j is the token number

Value

A data frame of Watt & Fabricius normalized formant tracks.

References

Watt, D., & Fabricius, A. (2002). Evaluation of a technique for improving the mapping of multiple speakers' vowel spaces in the F1 ~ F2 plane. Leeds Working Papers in Linguistics and Phonetics, 9, 159–173.

Examples

```
library(tidynorm)
library(dplyr)
ggplot2_inst <- require(ggplot2)

track_subset <- speaker_tracks |>
  filter(
    .by = c(speaker, id),
    if_all(
      F1:F3,
      .fns = \(x) mean(is.finite(x)) > 0.9
    ),
    row_number() %% 2 == 1
  )

track_norm <- track_subset |>
  norm_track_wattfab(
    F1:F3,
    .by = speaker,
    .token_id_col = id,
    .time_col = t,
    .drop_orig = TRUE
  )

if (ggplot2_inst) {
  track_norm |>
    ggplot(
      aes(F2_wf, F1_wf, color = speaker)
    ) +
    stat_density_2d(bins = 4) +
    scale_x_reverse() +
    scale_y_reverse() +
    scale_color_brewer(palette = "Dark2") +
    coord_fixed()
}


# returning the DCT coefficients
track_norm_dct <- track_subset |>
  norm_track_wattfab(
    F1:F3,
    .by = speaker,
```

```
    .token_id_col = id,
    .time_col = t,
    .drop_orig = TRUE,
    .return_dct = TRUE,
    .names = "{.formant}_wf"
  )

track_norm_means <- track_norm_dct |>
  summarise(
    .by = c(speaker, vowel, .param),
    across(
      ends_with("_wf"),
      mean
    )
  ) |>
  reframe_with_idct(
    ends_with("_wf"),
    .by = speaker,
    .token_id_col = vowel,
    .param_col = .param
  )


if (ggplot2_inst) {
  track_norm_means |>
    ggplot(
      aes(F2_wf, F1_wf, color = speaker)
    ) +
    geom_path(
      aes(
        group = interaction(speaker, vowel)
      )
    ) +
    scale_x_reverse() +
    scale_y_reverse() +
    scale_color_brewer(palette = "Dark2") +
    coord_fixed()
}
```

`norm_wattfab` *Watt & Fabricius Normalize*

Description

Watt & Fabricius Normalize

Usage

```
norm_wattfab(
  .data,
```

```
  ...,
  .by = NULL,
  .by_formant = TRUE,
  .agg_by = NULL,
  .agg_fun = function(x) mean(x, na.rm = T),
  .drop_orig = FALSE,
  .keep_params = FALSE,
  .names = "{.formant}_wf",
  .silent = opt("tidynorm.silent")
)
```

Arguments

.data	A data frame containing vowel formant data
...	<tidy-select> One or more unquoted expressions separated by commas. These should target the vowel formant data columns.
.by	<tidy-select> A selection of columns to group by. Typically a column of speaker IDs.
.by_formant	Ignored by this procedure
.agg_by	<tidy-select> A selection of columns to to pre-aggregate by. Typically a column of vowel category labels.
.agg_fun	A function to use to pre-aggregate values.
.drop_orig	Whether or not to drop the original formant data columns.
.keep_params	Whether or not to keep the Location (*_.L) and Scale (*_.S) normalization parameters
.names	A glue::glue() expression for naming the normalized data columns. The "{.formant}" portion corresponds to the name of the original formant columns.
.silent	Suppress normalization information messages when running a norm_*() function. (Defaults to FALSE, overwritable using option 'tidynorm.silent' or environment variable 'R_TIDYNORM_TIDYNORM_SILENT')

Details

This is a modified version of the Watt & Fabricius Method. The original method identified point vowels over which F1 and F2 centroids were calculated. The procedure here just identifies centroids by taking the mean of all formant values.

$$\hat{F}_{ij} = \frac{F_{ij}}{S_i}$$

$$S_i = \frac{1}{N}\sum_{j=1}^{N} F_{ij}$$

Where

- $\hat{F}$ is the normalized formant

- i is the formant number
- j is the token number

Pre-aggregation:

To address imbalanced data across vowel classes, you can pre-aggregate formant values by passing vowel class column(s) to `.agg_by`, and and an aggregating function to `.agg_fun`. The aggregation will be done after any `.pre_trans` transformation is applied, and will be done either formant intrinsically or formant extrinsically based on the setting passed to `.by_formant`.

Value

A data fame of Watt & Fabricius normalized formant values.

References

Watt, D., & Fabricius, A. (2002). Evaluation of a technique for improving the mapping of multiple speakers' vowel spaces in the F1 ~ F2 plane. Leeds Working Papers in Linguistics and Phonetics, 9, 159–173.

Examples

```
library(tidynorm)
ggplot2_inst <- require(ggplot2)

speaker_data_wattfab <- speaker_data |>
  norm_wattfab(
    F1:F3,
    .by = speaker,
    .names = "{.formant}_wf"
  )

if (ggplot2_inst) {
  ggplot(
    speaker_data_wattfab,
    aes(
      F2_wf,
      F1_wf,
      color = speaker
    )
  ) +
    stat_density_2d(
      bins = 4
    ) +
    scale_color_brewer(
      palette = "Dark2"
    ) +
    scale_x_reverse() +
    scale_y_reverse() +
    coord_fixed()
}
```

`options` *tidynorm Options*

Description

Options to control the verbosity of tidynorm functions. The convenience function `tidynorm_options()` will set these options within your current session. For a common behavior across R sessions, set the environment variables as described below. If you've silenced informational messages and want to double check what normalization steps have been taken, use `check_norm()`.

Checking Option Values

Option values specific to `tidynorm` can be accessed by passing the package name to `env`.

```
options::opts(env = "tidynorm")

options::opt(x, default, env = "tidynorm")
```

Options

tidynorm.silent Suppress normalization information messages when running a `norm_*()` function.

default: `FALSE`

option: tidynorm.silent

envvar: R_TIDYNORM_TIDYNORM_SILENT (evaluated if possible, raw string otherwise)

tidynorm.warnings Print warnings from tidynorm functions.

default: `TRUE`

option: tidynorm.warnings

envvar: R_TIDYNORM_TIDYNORM_WARNINGS (evaluated if possible, raw string otherwise)

See Also

options getOption Sys.setenv Sys.getenv

`reframe_with_dct` *Reframe with DCT*

Description

Reframe data columns using the Discrete Cosine Transform

Usage

```
reframe_with_dct(
  .data,
  ...,
  .token_id_col = NULL,
  .by = NULL,
  .time_col = NULL,
  .order = 5
)
```

Arguments

`.data`	A data frame
`...`	<tidy-select> One or more unquoted expressions separated by commas. These should target the vowel formant.
`.token_id_col`	<tidy-select> The token ID column.
`.by`	<tidy-select> A grouping column.
`.time_col`	A time column.
`.order`	The number of DCT parameters to return. If `NA`, all DCT parameters will be returned.

Details

This function will tidily apply the Discrete Cosine Transform with forward normalization (see dct for more info) to the targeted columns.

Identifying tokens:
The DCT only works on a by-token basis, so there must be a column that uniquely identifies (or, in combination with a `.by` grouping, uniquely identifies) each individual token. This column should be passed to `.token_id_col`.

Order:
The number of DCT coefficients to return is defined by `.order`. The default value is 5. Larger numbers will lead to less smoothing when the Inverse DCT is applied (see idct). Smaller numbers will lead to more smoothing.
If `NA` is passed to `.order`, all DCT parameters will be returned, which when the Inverse DCT is supplied, will completely reconstruct the original data.

Sorting by Time:
An optional `.time_col` can also be defined to ensure that the data is correctly arranged by time.

Value

A data frame with with the targeted DCT coefficients, along with two additional columns

.param The nth DCT coefficient number

.n The number of original data values

Examples

```
library(tidynorm)
library(dplyr)

speaker_small <- filter(
  speaker_tracks,
  id == 0
)

speaker_dct <- reframe_with_dct(
  speaker_small,
  F1:F3,
  .by = speaker,
  .token_id_col = id,
  .time_col = t
)

head(
  speaker_dct
)
```

---

reframe_with_dct_smooth

Reframe with DCT Smooth

Description

Apply a DCT Smooth to the targeted data

Usage

```
reframe_with_dct_smooth(
  .data,
  ...,
  .token_id_col,
  .by = NULL,
  .time_col = NULL,
  .order = 5,
  .rate = FALSE,
  .accel = FALSE
)
```

Arguments

.data	A data frame
...	<tidy-select> One or more unquoted expressions separated by commas. These should target the vowel formant.
.token_id_col	<tidy-select> The token ID column.
.by	<tidy-select> A grouping column.
.time_col	A time column.
.order	The number of DCT parameters to return. If NA, all DCT parameters will be returned.
.rate	Whether or not to include the rate of change of signal.
.accel	Whether or not to include acceleration of signal.

Details

This is roughly equivalent to applying reframe_with_dct followed by reframe_with_idct. As long as the value passed to .order is less than the length of the each token's data, this will result in a smoothed version of the data.

Identifying tokens:

The DCT only works on a by-token basis, so there must be a column that uniquely identifies (or, in combination with a .by grouping, uniquely identifies) each individual token. This column should be passed to .token_id_col.

Order:

The number of DCT coefficients to return is defined by .order. The default value is 5. Larger numbers will lead to less smoothing when the Inverse DCT is applied (see idct). Smaller numbers will lead to more smoothing.

If NA is passed to .order, all DCT parameters will be returned, which when the Inverse DCT is supplied, will completely reconstruct the original data.

Sorting by Time:

An optional .time_col can also be defined to ensure that the data is correctly arranged by time.

Additionally, if .time_col is provided, the original time column will be included in the output

Value

A data frame where the target columns have been smoothed using the DCT, as well as the signal rate of change and acceleration, if requested.

Examples

```
library(tidynorm)
library(dplyr)

ggplot2_inst <- require(ggplot2)

speaker_small <- filter(
```

```
  speaker_tracks,
  id == 0
)

speaker_dct_smooth <- speaker_small |>
  reframe_with_dct_smooth(
    F1:F3,
    .by = speaker,
    .token_id_col = id,
    .time_col = t,
    .order = 5
  )

if (ggplot2_inst) {
  speaker_small |>
    ggplot(
      aes(
        t, F1
      )
    ) +
    geom_point() +
    facet_wrap(
      ~speaker,
      scales = "free_x",
      ncol = 1
    ) +
    labs(
      title = "Original Data"
    )
}

if (ggplot2_inst) {
  speaker_dct_smooth |>
    ggplot(
      aes(
        t, F1
      )
    ) +
    geom_point() +
    facet_wrap(
      ~speaker,
      scales = "free_x",
      ncol = 1
    ) +
    labs(
      title = "Smoothed Data"
    )
}
```

`reframe_with_idct` *Reframe with IDCT*

Description

Reframe data columns using the Inverse Discrete Cosine Transform

Usage

```
reframe_with_idct(
  .data,
  ...,
  .token_id_col = NULL,
  .by = NULL,
  .param_col = NULL,
  .n = 20,
  .rate = FALSE,
  .accel = FALSE
)
```

Arguments

.data	A data frame
...	<tidy-select> One or more unquoted expressions separated by commas. These should target the vowel formant.
.token_id_col	<tidy-select> The token ID column.
.by	<tidy-select> A grouping column.
.param_col	A column identifying the DCT parameter number
.n	The size of the outcome of the IDCT
.rate	Whether or not to include the rate of change of signal.
.accel	Whether or not to include acceleration of signal.

Details

This will apply the Inverse Discrete Cosine Transform to the targeted columns. See idct.

Identifying tokens:

The IDCT only works on a by-token basis, so there must be a column that uniquely identifies (or, in combination with a .by grouping, uniquely identifies) each individual token. This column should be passed to .token_id_col.

Size of the output:

The output of the IDCT can be arbitrarily long as defined by the .n argument. .n can either be an integer, or an unqoted data column.

The Parameter Column:

The order of the DCT parameters is crucially important. The optional .param_col will ensure the data is properly arranged.

Value

A data frame with the IDCT of the targeted columns along with an additional `.time` column.

.time A column from 1 to `.n` by token

Examples

```
library(tidynorm)
library(dplyr)
ggplot2_inst <- require(ggplot2)

speaker_small <- filter(
  speaker_tracks,
  id == 0
)

speaker_dct <- speaker_small |>
  reframe_with_dct(
    F1:F3,
    .by = speaker,
    .token_id_col = id,
    .time_col = t,
    .order = 5
  )

speaker_idct <- speaker_dct |>
  reframe_with_idct(
    F1:F3,
    .by = speaker,
    .token_id_col = id,
    .param_col = .param,
    .n = 20
  )

if (ggplot2_inst) {
  speaker_small |>
    mutate(
      .by = c(speaker, id),
      time_index = row_number()
    ) |>
    ggplot(
      aes(
        time_index, F1
      )
    ) +
    geom_point() +
    labs(
      title = "Original Data"
    )
}

if (ggplot2_inst) {
```

```
  speaker_idct |>
    ggplot(
      aes(
        .time, F1
      )
    ) +
    geom_point() +
    labs(
      title = "DCT Smooth Data"
    )
}
```

`speaker_data` *Speaker Data*

Description

Speaker Data

Usage

```
speaker_data
```

Format

`speaker_data`**:**
A data frame with 10,697 rows and 8 columns

speaker Speaker ID column
vowel CMU Dictionary vowel class
plt_vclass Modified Labov-Trager vowel class
ipa_vclas IPA-like vowel class
word Word that the vowel appeared in
F1, F2, F3 The first, second and third formants, in Hz

`speaker_tracks` *Speaker Tracks*

Description

Speaker Tracks

Usage

```
speaker_tracks
```

Format

`speaker_tracks`:
A data frame with 20,000 rows and 9 columns

speaker Speaker ID column
id Within speaker id for each token
vowel CMU Dictionary vowel class
plt_vclass Modified Labov-Trager vowel class
ipa_vclas IPA-like vowel class
word Word that the vowel appeared in
t Measurement time point
F1, F2, F3 The first, second and third formants, in Hz

`tidynorm_options` *Set tidynorm options*

Description

Set tidynorm verbosity

Usage

```
tidynorm_options(
  .silent = opt("tidynorm.silent"),
  .warnings = opt("tidynorm.warnings")
)
```

Arguments

`.silent`	Suppress normalization information messages when running a `norm_*()` function. (Defaults to `FALSE`, overwritable using option 'tidynorm.silent' or environment variable 'R_TIDYNORM_TIDYNORM_SILENT')
`.warnings`	Print warnings from tidynorm functions. (Defaults to `TRUE`, overwritable using option 'tidynorm.warnings' or environment variable 'R_TIDYNORM_TIDYNORM_WARNINGS')

See Also

options

Examples

```
tidynorm_options(.silent = TRUE, .warnings = FALSE)

speaker_data |>
  norm_generic(F1:F3) ->
  norm1
```

```
tidynorm_options(.silent = FALSE, .warnings = TRUE)

speaker_data |>
  norm_generic(F1:F3) ->
  norm2
```

Index

∗ **datasets**
    speaker_data, 68
    speaker_tracks, 68

bark_to_hz, 3

check_norm, 4
check_norm(), *61*

dct, *7, 8*, *18*, *62*
dct_basis, 4

glue::glue(), *10*, *12*, *15*, *18*, *21*, *24*, *27*, *29*, *32*, *34*, *36*, *39*, *42*, *46*, *50*, *53*, *56*, *59*

hz_to_bark, 5, *10*, *12*, *39*
hz_to_mel, 6

idct, *62*, *64*, *66*
idct_accel, 7
idct_rate, 8

mel_to_hz, 9

norm_barkz, 10, *32*
norm_dct_barkz, 12, *18*
norm_dct_deltaF, 14, *18*
norm_dct_generic, 17, *18*
norm_dct_lobanov, *18*, 20
norm_dct_nearey, *18*, 23
norm_dct_wattfab, *18*, 26
norm_deltaF, 28, *32*
norm_generic, *18*, 31, *32*, *46*
norm_lobanov, *32*, 33
norm_nearey, *32*, 35
norm_track_barkz, 38, *46*
norm_track_deltaF, 41, *46*
norm_track_generic, 44, *46*
norm_track_lobanov, *46*, 49
norm_track_nearey, *46*, 52
norm_track_wattfab, *46*, 55
norm_wattfab, *32*, 58

options, 61, *69*

reframe_with_dct, 62, *64*
reframe_with_dct_smooth, 63
reframe_with_idct, *64*, 65

speaker_data, 68
speaker_tracks, 68

tidynorm_options, 69
tidynorm_options(), *61*